Creación de Aplicaciones Python

Lección 12: Dash

eip ESCUELA INTERNACIONAL POSGRADOS

ÍNDICE

Lección 12: Dash

PRESENTACIÓN Y OBJETIVOS

En esta Lección aprenderemos unas cuantas cosas. Siendo una de ellas el propio Framework Dash.

Algo que podríamos hacer es utilizar Dash sobre Jupyter Notebook,

Para lo cual haríamos lo siguiente: “pip install jupyter-dash”, no obstante, no trabajaremos con ello en esta ocasión.

Haremos Scripts en el propio ATOM.

Del mismo modo, se podría hacer algo más sofisticado y centrado en Big Data, por lo que es posible que se pueda profundizar más en otra Asignatura.

Por el momento, esta Introducción va a ser lo más completa posible para tener una idea de las cualidades que aporta Dash.

Objetivos

- *Aprender algunos conceptos básicos de Plotly*
- *Obtención de datos de la Web*
- *Hacer algunas pequeñas aplicaciones con Dash*

1. DIV EN UNA WEB

Algo que no hemos explicado para no extendernos más es la separación en <header>, <footer> etc. en una web.

No obstante lo que sí sería conveniente es explicar "div".

Div nos permite en HTML hacer divisiones.

En el siguiente ejemplo añadiremos 2 divisiones con 3 textos en cada división.

prueba_div.html

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
    <head>
        <meta charset="utf-8">
        <title>"Ejemplo con div"</title>
    </head>
    <body>
        <h1>Haremos 2 divisiones</h1>
        <div style="background-color:grey; color: black; font-style:normal">
            <p>División 1 - Texto 1</p>
            <p>División 1 - Texto 2</p>
            <p>División 1 - Texto 3</p>
        </div>
        <div style="background-color:blue; color: white; font-style:italic">
            <p>División 2 - Texto 1</p>
            <p>División 2 - Texto 2</p>
            <p>División 2 - Texto 3</p>
        </div>
    </body>
</html>
```

Figura 1.1: Explicación del "div" (parte 1)

Cuyo resultado debería ser algo similar al resultado siguiente.

Para hacerlo más fácil hemos añadido el estilo en el propio HTML. No hemos colocado dentro del "style" el último ";" y vemos que lo detecta igualmente. Pero los que se encuentran en medio es necesario añadirles.

Figura 1.2: Explicación del "div" (parte 2)

De modo que se ven claramente 2 divisiones.

Obviamente, es un ejemplo, el cual ha tratado de ser visual, pero no tendría mucho sentido en la práctica.

2. Obtención de datos de la web (APIs, etc).

Para explicar este punto nos iremos nuevamente a Jupyter Notebook.

Lo siguiente son pantallazos obtenidos del propio ejercicio.

Comencemos!

-1- Motivos de explicar este contenido

No es objetivo de esta asignatura obtener datos de la web

Pero tendremos algún ejercicio en el cual lo necesitaremos utilizar

Para simplificar el proceso de aprendizaje usaremos herramientas "sencillas"

Explicaremos:

- 2 posibles formas de obtener datos de una web
- .head() y .tail() de un DataFrame
- type(df)
- Ploteo simple con pandas
- Mencionaremos matplotlib para la gráfica con pandas

Figura 2.1: Obtención de Datos de Webs (parte 1)

-2- Obtención mediante "qandl"

In [1]:
```
# Lugar de donde obtenemos información con Python
# https://www.quandl.com/tools/python
```

In [2]:
```
import quandl
```

In [3]:
```
# Vamos a extraer las cotizaciones de Google en las fechas mencionadas

# Años 2015-2016-2017-2018

# formato: AÑO-MES-DÍA
```

In [4]:
```
data = quandl.get("WIKI/GOOGL", start_date="2015-1-1", end_date="2018-12-31")
```

Figura 2.2: Obtención de Datos de Webs (parte 2)

```
In [5]: # 5 primeras filas
        data.head()
```

Out[5]:

Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume
2015-01-02	532.60	535.8000	527.88	529.55	1327870.0	0.0	1.0	532.60	535.8000	527.88	529.55	1327870.0
2015-01-05	527.15	527.9899	517.75	519.46	2059119.0	0.0	1.0	527.15	527.9899	517.75	519.46	2059119.0
2015-01-06	520.50	521.2100	505.55	506.64	2731813.0	0.0	1.0	520.50	521.2100	505.55	506.64	2731813.0
2015-01-07	510.95	511.4900	503.65	505.15	2345875.0	0.0	1.0	510.95	511.4900	503.65	505.15	2345875.0
2015-01-08	501.51	507.5000	495.02	506.91	3662224.0	0.0	1.0	501.51	507.5000	495.02	506.91	3662224.0

```
In [6]: # 5 últimas filas
        data.tail()
```

Out[6]:

Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume
2018-03-21	1092.57	1108.70	1087.21	1094.00	1990515.0	0.0	1.0	1092.57	1108.70	1087.21	1094.00	1990515.0
2018-03-22	1080.01	1083.92	1049.64	1053.15	3418154.0	0.0	1.0	1080.01	1083.92	1049.64	1053.15	3418154.0
2018-03-23	1051.37	1066.78	1024.87	1026.55	2413517.0	0.0	1.0	1051.37	1066.78	1024.87	1026.55	2413517.0
2018-03-26	1050.60	1059.27	1010.58	1054.09	3272409.0	0.0	1.0	1050.60	1059.27	1010.58	1054.09	3272409.0
2018-03-27	1063.90	1064.54	997.62	1006.94	2940957.0	0.0	1.0	1063.90	1064.54	997.62	1006.94	2940957.0

Figura 2.3: Obtención de Datos de Webs (parte 3)

```
In [7]: # Imaginemos que solo queremos el precio de cierre de la acción

In [8]: data = data[["Close"]]
        print(data)
```

```
              Close
Date
2015-01-02   529.55
2015-01-05   519.46
2015-01-06   506.64
2015-01-07   505.15
2015-01-08   506.91
...             ...
2018-03-21  1094.00
2018-03-22  1053.15
2018-03-23  1026.55
2018-03-26  1054.09
2018-03-27  1006.94

[813 rows x 1 columns]
```

Figura 2.4: Obtención de Datos de Webs (parte 4)

In [9]:
```
# Incluso existe la forma de hacerlo añadiendo el número de columna.
# Columna 4 en este caso.
# OJO, Normalmente los números empiezan en 0 al contar filas ó columnas
df = quandl.get("WIKI/GOOGL.4", start_date="2015-1-1", end_date="2018-12-31")
print(df)
```

```
              Close
Date
2015-01-02   529.55
2015-01-05   519.46
2015-01-06   506.64
2015-01-07   505.15
2015-01-08   506.91
...             ...
2018-03-21  1094.00
2018-03-22  1053.15
2018-03-23  1026.55
2018-03-26  1054.09
2018-03-27  1006.94

[813 rows x 1 columns]
```

In [10]:
```
type(df)
```

Out[10]:
```
pandas.core.frame.DataFrame
```

In [11]:
```
type(data)
```

Out[11]:
```
pandas.core.frame.DataFrame
```

Figura 2.5: Obtención de Datos de Webs (parte 5)

```
In [12]: # Es un pandas..DataFrame (en ambos casos)

In [13]: # https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.html

In [14]: # por defecto un gráfico de líneas
         data.plot()

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x1e7dbdc8978>

In [15]: # si queremos quitar ese:
         # <matplotlib.axes..

         import matplotlib.pyplot as plt

         data.plot()
         plt.show()
```

Figura 2.6: Obtención de Datos de Webs (parte 6)

```
In [16]: # Y si queremos añadir un título y unos ejes

         data.plot(title="Gráfica de cotización de Google",
                   xlabel = "2015-2018",
                   ylabel = "Cotización en $ USD" )

         plt.show()
```

Figura 2.7: Obtención de Datos de Webs (parte 7)

-3- Obtención mediante "yfinance"

```
In [17]: # (OTRA FORMA DE OBTENER LA INFORMACIÓN)

In [18]: # https://pypi.org/project/yfinance/
         # Del Link anterior obtenemos esta información.
         # pip install yfinance

In [19]: # ModuleNotFoundError: No module named 'yfinance'

In [20]: # !pip install yfinance

In [21]: import yfinance as yf

In [22]: df2 = yf.download("GOOGL", "2015-1-1", "2018-12-31")
         [**********100%**********]  1 of 1 completed

In [23]: type(df2)
Out[23]: pandas.core.frame.DataFrame
```

Figura 2.8: Obtención de Datos de Webs (parte 8)

```
In [24]: df2.head()
```

Out[24]:

Date	Open	High	Low	Close	Adj Close	Volume
2014-12-31	537.739990	538.400024	530.200012	530.659973	530.659973	1232400
2015-01-02	532.599976	535.799988	527.880005	529.549988	529.549988	1324000
2015-01-05	527.150024	527.989990	517.750000	519.460022	519.460022	2059100
2015-01-06	520.500000	521.210022	505.549988	506.640015	506.640015	2722800
2015-01-07	510.950012	511.489990	503.649994	505.149994	505.149994	2345900

```
In [25]: df2.tail()
```

Out[25]:

Date	Open	High	Low	Close	Adj Close	Volume
2018-12-21	1032.040039	1037.670044	981.190002	991.250000	991.250000	5232800
2018-12-24	984.320007	1012.119995	977.659973	984.669983	984.669983	1818000
2018-12-26	997.989990	1048.449951	992.650024	1047.849976	1047.849976	2315900
2018-12-27	1026.199951	1053.339966	1007.000000	1052.900024	1052.900024	2299800
2018-12-28	1059.500000	1064.229980	1042.000000	1046.680054	1046.680054	1719900

Figura 2.9: Obtención de Datos de Webs (parte 9)

In [26]:
```
df2 = df2[["Close"]]
df2.head()
```

Out[26]:

Date	Close
2014-12-31	530.659973
2015-01-02	529.549988
2015-01-05	519.460022
2015-01-06	506.640015
2015-01-07	505.149994

In [27]:
```
data.head()
```

Out[27]:

Date	Close
2015-01-02	529.55
2015-01-05	519.46
2015-01-06	506.64
2015-01-07	505.15
2015-01-08	506.91

Figura 2.10: Obtención de Datos de Webs (parte 10)

```
In [28]: df2.tail()
```

Out[28]:

Date	Close
2018-12-21	991.250000
2018-12-24	984.669983
2018-12-26	1047.849976
2018-12-27	1052.900024
2018-12-28	1046.680054

```
In [29]: data.tail()
```

Out[29]:

Date	Close
2018-03-21	1094.00
2018-03-22	1053.15
2018-03-23	1026.55
2018-03-26	1054.09
2018-03-27	1006.94

Figura 2.11: Obtención de Datos de Webs (parte 11)

```
In [30]: df2.plot(title="Gráfica de cotización de Google",
                  xlabel = "2015-2018",
                  ylabel = "Cotización en $ USD" )

         plt.show()
```

Figura 2.12: Obtención de Datos de Webs (parte 12)

```
In [31]: data.plot(title = "Gráfica de cotización de Google",
                  xlabel = "2015-2018",
                  ylabel = "Cotización en $ USD" )

         plt.show()
```

Figura 2.13: Obtención de Datos de Webs (parte 13)

3. PLOTLY PARA OBTENCIÓN DE IRIS DATASET

Lo primero que podríamos comentar es que podemos hacer uso del Iris Dataset por medio de la librería plotly, de la siguiente manera:

dash_1.py

```
1  import plotly.express as px
2
3  df = px.data.iris()
4  print(df)
```

Figura 3.1: Obtención del Iris DataSet con Plotly

El cual tenemos a continuación, y debemos fijarnos en la forma de nombrar las columnas (sepal_length por ejemplo).

Python - dash_1.py:6 ✔

	sepal_length	sepal_width	...	species	species_id
0	5.1	3.5	...	setosa	1
1	4.9	3.0	...	setosa	1
2	4.7	3.2	...	setosa	1
3	4.6	3.1	...	setosa	1
4	5.0	3.6	...	setosa	1
..	...	...	...	...	...
145	6.7	3.0	...	virginica	3
146	6.3	2.5	...	virginica	3
147	6.5	3.0	...	virginica	3
148	6.2	3.4	...	virginica	3

Figura 3.2: Iris DataSet

4. DASH: INSTALACIÓN

Lo primero es irse a la siguiente web, donde encontramos información acerca de la instalación.

https://dash.plotly.com/installation

Para instalar Dash, nos dice (en este momento): pip install dash==1.19.0

lo escribimos en la CMD y pulsamos Enter.

También tenemos esta opción, "plotly"

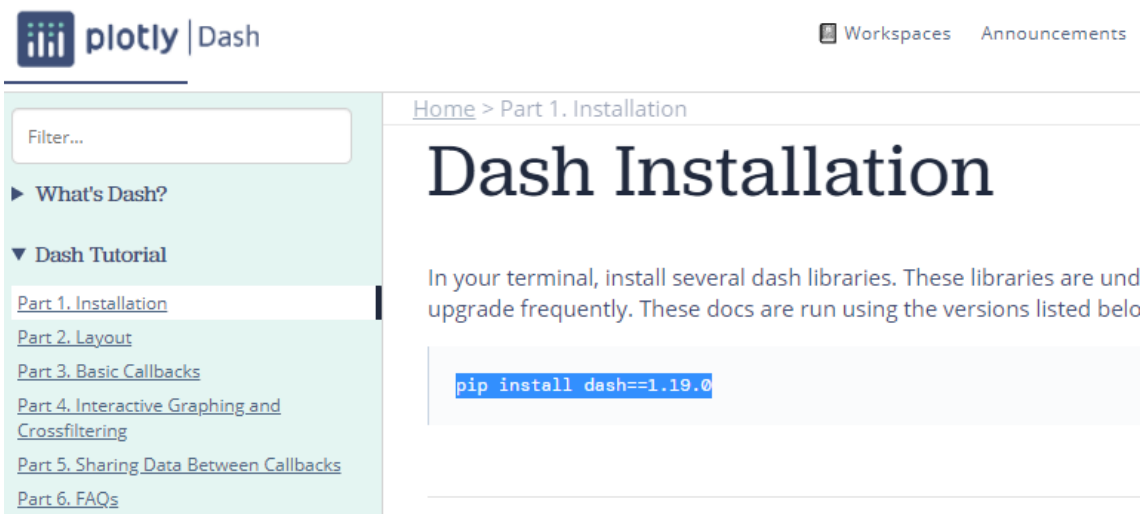

Figura 4.1: Instalación de Dash

5. DASH: PRIMERA APLICACIÓN DASH

Lo siguiente que podríamos hacer es nuestra primera aplicación Dash.

En la misma vemos cómo podemos con pocas líneas crear la estructura de la misma.

La parte inferior es la ejecución en la "cmd", de modo que deberemos copiar esa ruta, y escribirla en el navegador.

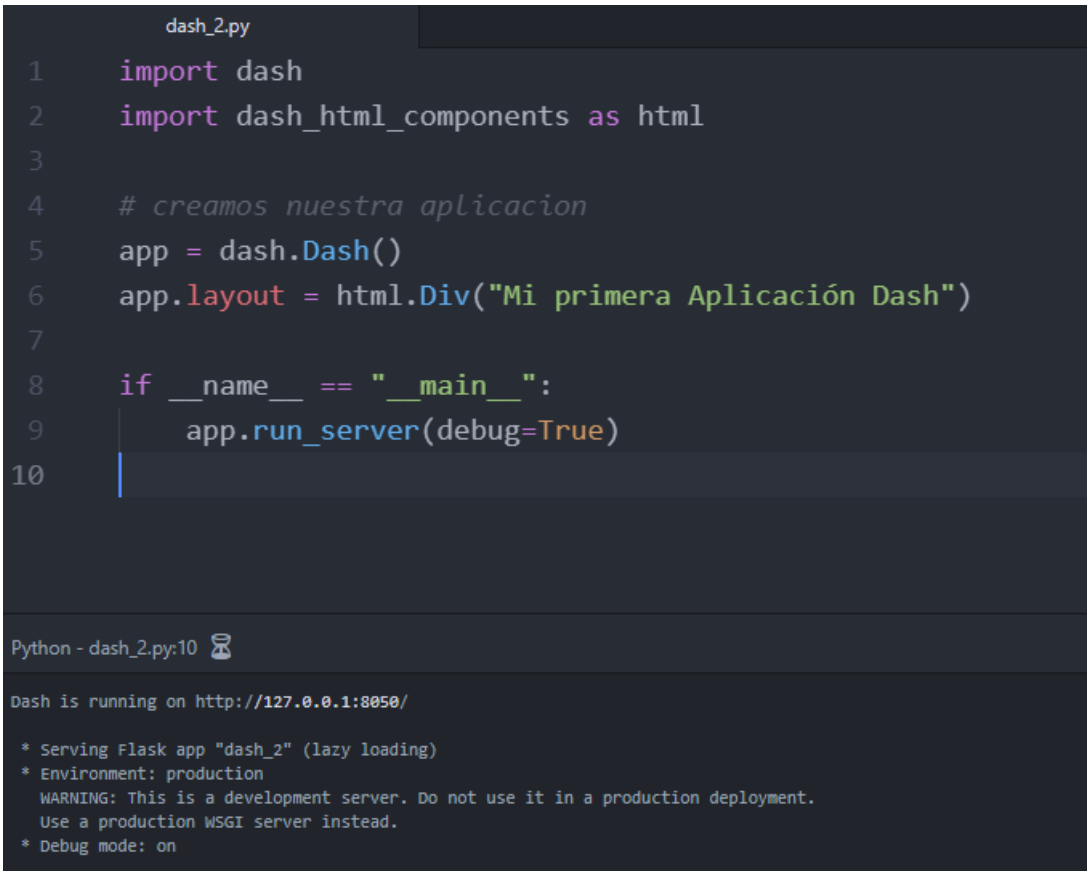

Figura 5.1: Primera Aplicación Dash (parte 1)

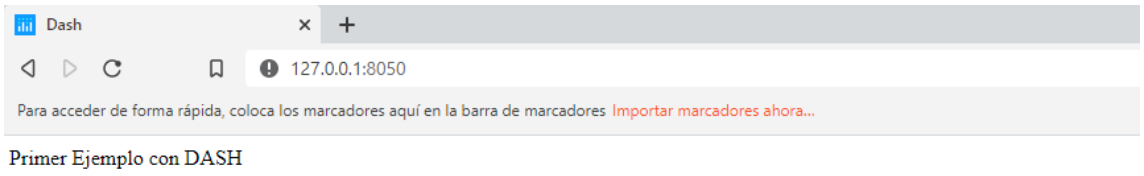

Figura 5.2: Primera Aplicación Dash (parte 2)

6. DASH: APLICACIÓN CON UNA GRÁFICA DEL IRIS DATASET

Vamos a hacer una Aplicación que obtenga una de las gráficas del Iris Dataset ya vistas:

dash_3.py

```
import dash
import dash_core_components as dcc
import dash_html_components as html
import plotly.express as px

df = px.data.iris()  # iris is a pandas DataFrame
figura = px.scatter(df, x="petal_length", y="petal_width", color="species")

app = dash.Dash()
app.layout = html.Div(children=[
    html.H1(children="Gráfica del Iris Dataset para el Pétalo (Largo-Ancho)"),
    dcc.Graph(figure=figura)])

if __name__ == "__main__":
    app.run_server(debug=True)
```

Figura 6.1: Iris Dataset (parte 1)

Y vemos que sale:

Figura 6.2: Iris Dataset (parte 2)

7. DASH: CÓMO CERRAR LAS APLICACIONES

Para poder cerrar una aplicación: CTRL + C (“cmd”)

Para todos aquellos casos en los que intentamos trabajar con una aplicación pero se nos abre la anterior, disponemos de otra opción, que es irnos al: “Administrador de Tareas”.

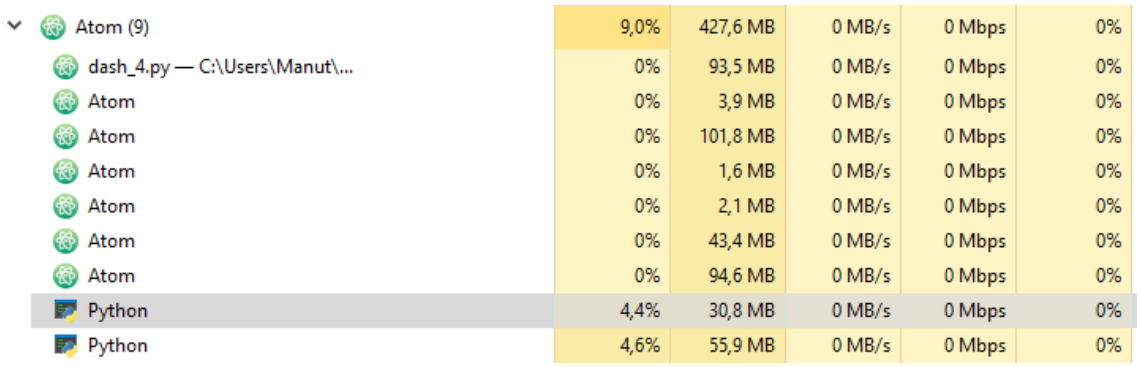

Figura 7.1: Cierre de las Aplicaciones (parte 1)

En aquellas donde dice Python le indicamos Finalizar tarea, si es que no hemos cerrado correctamente la ejecución de las aplicaciones.

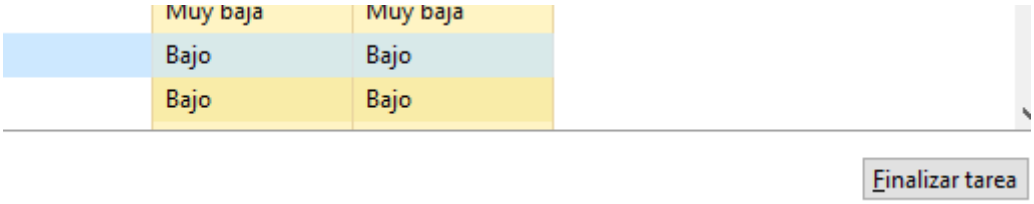

Figura 7.2: Cierre de las Aplicaciones (parte 2)

Entre unas aplicaciones y otras con Dash será conveniente cerrar las tareas, ya que de lo contrario es posible que no se abra la aplicación más reciente.

8. Dash: Actualización HTML en Tiempo Real

Algo que podemos hacer en Dash es actualizar dinámicamente elementos HTML en tiempo real, sin necesidad de actualizar la página.

Para este ejemplo haremos algo simple: un campo de entrada y uno de salida.

Para este simple ejemplo tomaremos la entrada y la repetiremos a la salida.

Estamos usando un "decorador".

Estamos, como vemos utilizando otras dependencias de Dash para hacer la parte de Input y de Output.

dash_4.py

```python
import dash
from dash.dependencies import Input, Output
import dash_core_components as dcc
import dash_html_components as html

app = dash.Dash()
app.layout = html.Div([
    dcc.Input(id="input", value="Escribe algo aquí", type="text"),
    html.Div(id="output")
 ])

@app.callback(
    Output(component_id="output", component_property="children"),
    [Input(component_id="input", component_property="value")]
)

def update_value(input_data):
    return 'Entrada: "{}"'.format(input_data)

if __name__ == "__main__":
    app.run_server(debug=True)
```

Figura 8.1: Actualización HTML en tiempo real (parte 1)

Cuyo resultado será:

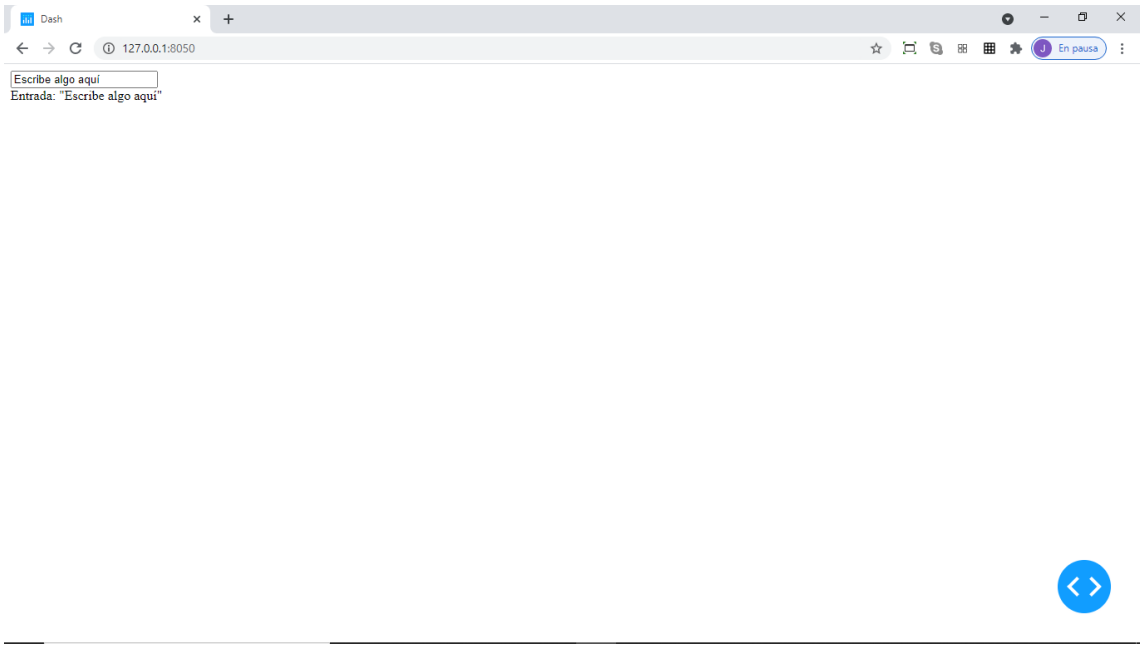

Figura 8.2: Actualización HTML en tiempo real (parte 2)

Que es escribimos algo en la entrada veremos cómo se actualiza la información.

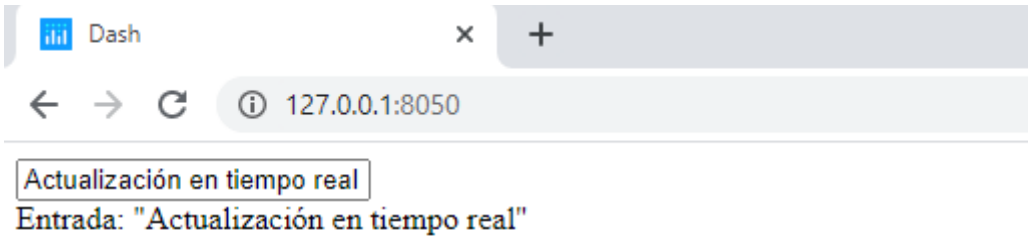

Figura 8.3: Actualización HTML en tiempo real (parte 3)

9. Dash: Gráficos para Mercados Financieros

Podríamos hacer una pequeña aplicación para Finanzas, que es una utilidad de Dash.

dash_5.py

```
import dash
import dash_core_components as dcc
import dash_html_components as html
import plotly.express as px
import quandl

df = quandl.get("WIKI/GOOGL.4", start_date="2015-1-1", end_date="2018-12-31")
# print(df)
figura = px.line(df, title="Cotización de Google: ENERO 2015 - DICIEMBRE 2018")

app = dash.Dash()
app.layout = html.Div(children=[
    html.H1(children="APLICACIÓN CON DASH PARA MERCADOS FINANCIEROS"),
    dcc.Graph(figure=figura)])

if __name__ == "__main__":
    app.run_server(debug=True)
```

Figura 9.1: Gráficos para Finanzas (parte 1)

Figura 9.2: Gráficos para Finanzas (parte 2)

10. PUNTOS CLAVE

- Nuevamente hemos visto que los conocimientos en HTML son útiles cuando trabajamos con Frameworks de Python para hacer Aplicaciones Web.

- Plotly nos permite hacer gráficas con Python de bastante buena calidad.

- Dash nos permite hacer Aplicaciones webs